[PCT/EP00/00386]

# METHOD FOR GRAPHICALLY REPRESENTING AND/OR PROCESSING VALUES OF DATA TYPES

## Background of the Invention

The present invention relates to a method for graphically representing and/or processing values of data types of a formally defined data structure existing as a value tree.

Programming and description languages require a formal syntax to describe data types and their values. Therefore, methods are known for defining data structures using a formal notation, independently of a specific language. Generally, the data types defined in accordance with one of these methods can be of any desired complexity and dynamically change in part. It is, therefore, necessary that such data types be clearly represented for display purposes or to enable user input of values. This is particularly necessary for the management of communications networks, since very complex data structures are often processed in these networks.

In most languages and syntax notations, rules already exist for textually representing values. Thus, for example, when working with ASN.1 data types in accordance with ITU-T recommendation X.208, the value is represented as a character string in a manner that is not very straightforward.

The structure of the data type is more easily recognized when the known form of representation as a tree is used. A representation of this kind is described in the firm publication "IBM TMN Development Platform", Piscataway, N.J., U.S., 1998. Another known method includes assigning the value of an attribute to a graphic component by manually creating rules, for example, defining the color of the graphic object

by the value of the data type, as described for example in the company publication, Objective Systems Integrators: "NetExpert Framework Overview", Folsom, CA, U.S. 1997.

Since any data type definitions at all are possible, the known methods expend a great deal of time programming a separate graphical user-interface window for each new data type.

The object of the present invention is to reduce this time expenditure and to, nevertheless, achieve a straightforward representation, which will enable the data structure to be tested and, if necessary, processed.

This objective is achieved by the method in accordance with the present invention in that

- a window is assigned as a graphical user interface to the data structure;
- generic, scalable, graphical user-interface components are inserted hierarchically in the window, the value tree of the data structure being mapped onto the user-interface components;
- the graphical user interface components are in a relation to the nodes of the value tree in a manner that is recognizable to the user; and
- that a graphical or textual representation of the value is selectable for each subtree of the value tree.

The clear, straightforward representation of a complex data type using the method in accordance with the present invention saves the user from having to search extensively for definitions and helps him to avoid errors when inputting values for this data type. In this context, the present invention offers the possibility of concealing redundant information and of presenting important information in detail, since in the method according to the present invention, each user has a chance to decide which information should be displayed and for which information a compact representation

suffices. In particular, the method according to the present invention allows a simple value assignment in the processing of the data types.

One further refinement of the present invention provides for an especially simple and secure value assignment to be carried out in that, for a processing of the value tree, a list of all values which are compatible with respect to assignment with the represented data type is derived for each node, and that, in each case, one value is selected from the list for a value assignment. In the process, to avoid input errors, it can be provided when compiling value lists, that the number of values to be accepted in the list be restricted in accordance with predefined rules, depending on the current context.

Another refinement of the present invention provides for a visualization of the window to be first undertaken at the time of an initialization of the graphical user interface and, after that, data, in particular value lists, to be initialized, which are derived for a processing. As a result of the faster display build-up associated therewith, the user can already obtain an overview of the data structure before all data required for displaying and processing the data type are initialized.

A data type can also be graphically displayed when no additional information, described as metadata, on the data type is available in the graphical interface application, when, in accordance with another refinement of the present invention, the value to be represented is transferred in a transfer syntax, which contains all necessary information for the representation with respect to the data type and the value assignment.

In accordance with one advantageous specific embodiment of the method according to the present invention, data types, whose exact type assignment can first be determined at the execution

time in accordance with the late binding principle, are inserted as a dynamically changeable subtree in the value tree represented by the graphical user interface. From this, one derives the advantage for these data types that the representation of the current value assignment does not first require opening subwindows, but can take place directly in the main window.

Another advantageous specific embodiment provides that, for data types whose exact type assignment is first defined in accordance with the late binding principle at the execution time by the marking of another node (for example, "ANY DEFINED BY" in the description language ASN.1), the user is prompted to input whether the assignment should be carried out automatically or following a manual input. Thus, an assignment is also possible when the information required for the automatic assignment is not available.

In accordance with another advantageous refinement, values can be transferred from one subtree into another by intermediately storing and clicking on the subtree in question. In the process, an assignment compatibility of the data types assigned to the subtrees is, in fact, necessary. However, it is not necessary for the data types to be instantiated within the same value tree.

Another advantageous refinement makes possible a simple use, together with other programs, and the simple mapping of the general overall identity in that the method is implemented by one or more program modules that can be integrated in the application programs.

Yet another advantageous refinement of the present invention provides that additional information to be displayed can be stored for each node of the value tree which can be uniquely named by the displayed type and the relation to the higher-order type. In this manner, additional text elements

can be displayed for specific data types, which are produced at any point in time following the program creation, and which can be dynamically integrated in the interface at the execution time.

Another embodiment of the present invention provides that it be continually checked during inputting of a value to determine whether the input value is permissible for the corresponding data type, and whether the input value is identical to the currently active value of the data type, and that the result be made known to the user. The purpose of this embodiment is to further enhance security and the speed attained in the processing of data types.

In addition, the display and/or the processing can be facilitated in that the display format can be altered already at the time that a value is input and, thus, for example, a numerical value is either displayed as a decimal or binary value, before a value is accepted into the value tree.

Exemplary embodiments of the present invention are represented by several figures in the drawing and are more closely explained in the following description. The figures show:

Figure 1 a window of a graphical user interface in accordance with the method of the present invention;

Figure 2 an example of the formal data definition in accordance with ASN.1; and

Figure 3 the principle of a generic, graphic representation.

In the form of a structured text, Figure 2 shows the definitions of the data types "GetArgument" and "BaseManagedObjectClass". The type "GetArgument" includes two components, "baseManagedObjectClass" and "baseManagedObjectInstance", the names being given to the

left, and the types of the components to the right, namely "BaseManagedObjectClass" and GraphicString". The type "BaseManagedObjectClass" is a CHOICE having likewise two components, namely "globalForm" of type "OBJECTIDENTIFIER" and "localForm" of type "INTEGER".

Figure 3 depicts the principle of a generic, graphic representation of the formal data definition illustrated in Figure 2. In this context, the value tree is shown to the left, while the graphic representation is indicated to the right. In this instance and in the text that follows, the abbreviation GUI (= Graphical User Interface) is also used for the graphical user interface. Solid lines running between the blocks signify that the underlying elements are contained in the node situated above them, while the dotted lines signify that each graphic representation is in relation to the particular data type.

Figure 1 illustrates the relevant graphical user interface as a window 1, in which all included data types are contained as graphical user-interface components. The root of the value tree (Figure 3) is indicated by the GetArgument SEQUENCE. Input bars 2, 3 and 4, having the labels "globalForm", "localForm" and "baseManagedObjectInstance" correspond to the leaf objects of the value or GUI tree. Buttons 5 through 9 can be used to select between a textual representation (minus) and a graphic representation (plus) for each node and each leaf object.

Other buttons 10, 11 (radio buttons) can be used to select between two data types to be alternatively processed (in this case, leaf objects). Input bars 2, 3, 4 are each provided with a button 12, 13, 14, which can be used to open a window 15 for selecting values stored as constants.

In addition, the graphical user interface has buttons 16, 17 for terminating the processing and storing the processed data,

as well as for exiting the graphical user interface without storing.

5